Machine Learning of SpamBase Dataset

Research

using this dataset:

https://archive.ics.uci.edu/ml/datasets/spambase

Code URL:

https://github.com/DogDogBird/Machine-Learning.git

Subject Areas:
Machine Learning - tensorflow

Keywords:
Spambase, MLP, Deep Learning, tensorflow

Author for correspondence:
Kyubin Kyoung
kyubin0704@gmail.com

The Usage of spambase Dataset to get Higher Accuracy and Precision

Kyubin Kyong[1]

While using e-mails, we each have spam mail boxes where the spams go. However sometimes, spam that should go into the spam mailbox goes to normal mailbox and the other applies too. To get more accuracy, spam goes to spam mailbox, ham goes to ham mailbox, using ML will be an answer. So far there exists a ML code in github that uses multinominal method.

1. BackGround

Today there are lots of small websites trying to get users such as gambling or adult site. Even the major companies need more users. The one way they are getting users is by sending e-mails. Users call those unpleasant mails spam mails.

There are so many datasets in UCI Website and there existed a dataset that handles information about spam mail. Consisted of the specific word frequency

[1] 3rd Grade. Kyunghee University student. Korea

especially like !, $, #, etc..

On the website, there are some codes that handles this spambase.csv dataset. They handle this data using multinominal method. They are just getting a result of accuracy. So the thing I tried to do is using MLP(Multi Layer Perceptron) with 4~6 hidden layers and get precision as well as accuracy.

2. Dataset Structure[i]

There are 58 rows. Data in 1~54 row means the frequency of the specific term.

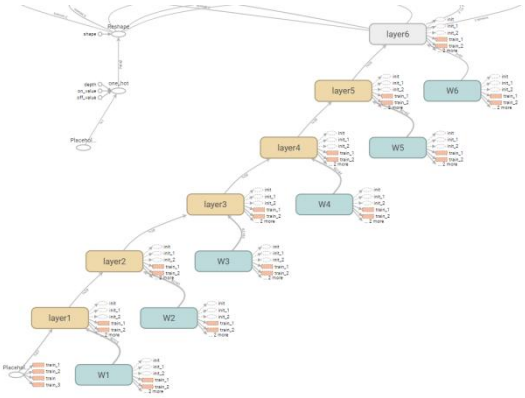
Data in 55~57 row means the average, maximum and the total value of the capital run_length_encoding.

The last row means if it is Spam or not.

However, there was a problem using this raw dataset. So the dataset needed normalization. Dividing data into the largest number of the dataset has been an answer. I divided 57*4601 data into 15841 and the data became normalized between 0 to 1.

The classes are divided into 2 section. Ham or Spam.

3. Why using MLP?



(Fig. 1) Layers in Tensor Board

MLP is abbreviation of Multi Layer Perceptron. It means as there are a neural network called perceptron, MLP is going to use multiple perceptrons. We call that hidden layer. So I really works like human neural network. And if there are lots of hidden layers this is called Deep Learning.

The reason I am using MLP is because this dataset is frozen. As there are 4601.labels, this number is not too big, and there are just two classes, I didn't need to use CNN.

There are 5 hidden layers and as my data vector is [57,2] I am using [57,28], four [28,28] and [28,2] layer.

Using relu, this program became much more trustable. And visualized the layers by tensorboard

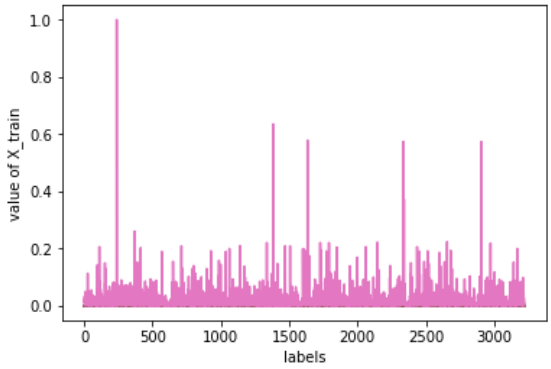


Fig 2. X_train Data

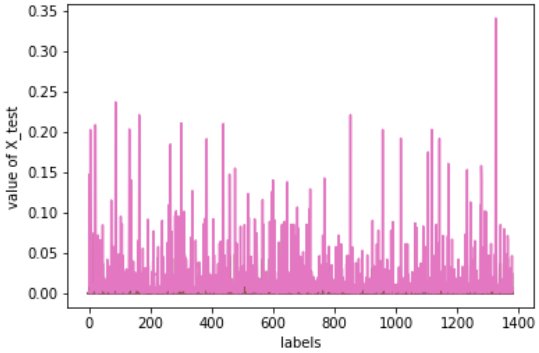


Fig 3, X_test Data

Training sets and test sets are splitted from one dataset. training sets handle 70% of the data and the remains are test sets. So there are 3220 training sets and 1381 test sets.

However using only this data caused an error. Cost became zero. Using one_hot_encoding was the solution to this problem.

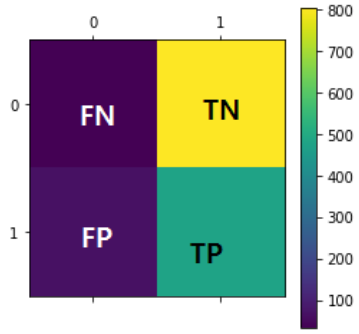
One hot encoding is a process by which categorical variables are converted into a form that could be provided to ML algorithms to do a better job in prediction.

As there are lots of optimizer algorithm, I chose the best working algorithm.

Optimizer	Accuracy
AdadeltaOptimizer	0.605358
AdagradOptimizer	0.605358
AdamOptimizer	0.917451
FtrlOptimizer	0.605358
ProximalGradientDescentOptimizer	0.605358

The AdamOptimizer does the best work making Accuracy up to 91%. So calculating precision with this optimizer I could get the confusion matrix.

```
TP 479.0
TN 804.0
FP 32.0
FN 66.0
```



(Fig. 4) confusion matrix

4. Result

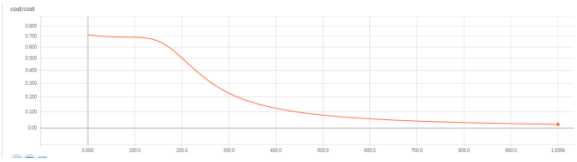


fig 5. cost

Cost gradiently descents while the time flows.

The precision is 0.937378 by the equation

Precision = TP / TP + FP

Naive Bayes

Method	Accuracy Avg	Accuracy Std	AUC Avg	AUC Std	Top 5 Features (y=ham)	Top 5 Features (y=spam)
Gaussian	0.80858	0.01097	0.85702	0.00859	[('650', 1.3194578005115096), ('credit', 1.2579437340153434), ('hpl', 0.9440358056259861), ('people', 0.537319693094629 12), ('george', 0.42955498721227592)]	[('credit', 2.2747826086956491), ('font', 1.3878418972332005), ('business', 0.5405454545454522), ('people', 0.5380316205533594), ('over', 0.5166719367588934 5)]
Multinomial	0.86884	0.00866	0.95108	0.00528	[('credit', 0.1439320882325051 7), ('650', 0.1435122978615089 3), ('hpl', 0.0948094057443072 4), ('people', 0.0566711455397133 77), ('font', 0.0482575926879947 53)]	[('credit', 0.2350035593514518 4), ('font', 0.1454085204752844 6), ('people', 0.0564197032951460 83), ('over', 0.0525073513813552 88), ('business', 0.0517208562902418 96)]
Bernoulli (alpha=1.0, bin=0.31)	0.87806	0.00736	0.95719	0.00401	[('credit', 0.1057740924259278 6), ('people', 0.0725048033168166 63), ('hpl', 0.0692688461927396 8), ('font', 0.0570330670441905 47), ('george', 0.0501567398119122 52)]	[('credit', 0.1190450352685837), ('font', 0.1069994574064025), ('people', 0.0775908844275636 78), ('3d', 0.0660879001627780 32), ('over', 0.0656538252848616 06)]

fig 6. data shown at

https://github.com/sampepose/SpamClassifier/blob/master/my_test.py

```
In [7]: spam_bayes = MultinomialNB()
        spam_bayes.fit(X_train, y_train)
Out[7]: MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True)

In [8]: spam_bayes.score(X_test, y_test)
Out[8]: 0.7920738327904451 9

In [9]: spam_bayes.score(X_train, y_train)
Out[9]: 0.7901413955636099 8
```

fig 7. data shown at

https://github.com/JonathanKross/spambase/blob/master/spamalot.ipynb

The accuracy also increased using MLP than using Multinominal such as in fig.5 and fig.6.

	test 1	test 2	test 3
Accuracy	0.876177	0.919623	0.926141
precision	0.77994	0.870307	0.890653
Recall	0.955963	0.93578	0.926606
test 4	test 5	test 6	test 7
0.908038	0.909486	0.905865	0.8979
0.877256	0.911765	0.863398	0.859431
0.891743	0.853211	0.904587	0.886239
test 8	test 9	test 10	Average
0.887763	0.923244	0.85735	0.901159
0.806604	0.883072	0.828302	0.857073
0.941284	0.92844	0.805505	0.902936

5. Conclusion

This is the best accuracy using the UCI spambase dataset. Using MLP is better than Multinominal.

However to get better result, we need more attribute and labels. Train more data, and testing will be much better. As a result we can defeat spam mails!

---

[i] Issued in readme.txt